Particle Filters for Real-Time Fault Detection in Planetary Rovers

Richard Dearden and Dan Clancy
Research Institute for Advanced Computer Science
NASA Ames Research Center
Mail Stop 269-3 Moffett Field, CA 94035 USA
Email:dearden,clancy@ptolemy.arc.nasa.gov

Abstract.

Planetary rovers provide a considerable challenge for artificial intelligence in that they must operate for long periods autonomously, or with relatively little intervention. To achieve this, they need to have on-board fault detection and diagnosis capabilities. Traditional model-based diagnosis techniques are not suitable for rovers due to the tight coupling between the vehicle's performance and its environment. Hybrid diagnosis using particle filters is presented as an alternative, and its strengths and weaknesses are examined. We also present some extensions to particle filters that are designed to make them more suitable for use in diagnosis problems.

1 Introduction

Planetary rovers provide a considerable challenge for artificial intelligence in that they must operate for long periods autonomously, or with relatively little intervention. To achieve this, they need (among other things) to have on-board fault detection and diagnosis capabilities in order to determine the actual state of the vehicle, and decide what actions are safe to perform. However, as we will discuss below, traditional approaches to diagnosis are unsuitable for rovers, and we must turn to hybrid approaches. In this paper we describe an approach to hybrid diagnosis based on *particle filters* [2, 7, 3]. We show that the characteristics of diagnosis problems present some difficulties for standard particle filters, and describe an approach for solving this problem. We will use rovers as a motivating example throughout this paper, but the techniques we describe can be applied to any hybrid diagnosis problem.

The diagnosis problem is to determine the current state of a system given a stream of observations of that system. In traditional model-based diagnosis systems such as Livingstone [14], diagnosis is performed by maintaining a set of candidate hypotheses (in Livingstone, a single hypothesis was used) about the current state of the system, and using the model to predict the expected future state of the system given each candidate. The predicted states are then compared with the observations of what actually occurred. If the observations are consistent with a particular state that is predicted, that state is kept as a candidate hypothesis. If they are inconsistent, the candidate is discarded. Traditional diagnosis systems typically use a logic-based representation, and use *monitors* to translate continuous-valued sensor readings into discrete-valued variables. The system can then reason about the discrete variables, and compare them with the predictions of the model using constraint propagation techniques.

Unlike the spacecraft domains that Livingstone has been applied in, rover performance depends significantly on environmental interactions. The on-board sensors provide streams of continuous valued data that varies due to noise, but also due to the interaction between the rover and its environment. For example, a rover may have a sensor that reports the current drawn by a wheel. In normal operation, this quantity may vary considerably, increasing when the vehicle is climbing a hill, and decreasing on downward slopes. The diagnosis system needs to be able to distinguish a change in the current drawn due to the terrain being traversed from a change due to a fault in the wheel. A second issue for rovers is that their weight and power is very tightly constrained. For this reason, any on-board diagnosis system must be computationally efficient, and should be able to adapt to variations in processor availability. Ideally, we would also like it to adapt based on its own performance, spending more time on diagnosis when a fault is likely to have occurred, and less time when the system appears to be operating normally.

A rover's close coupling with its environment poses a considerable problem for diagnosis systems that use discrete models. A particular sensor reading may be normal under certain environmental conditions, but indicative of a fault in others, so any monitor that translates the sensor reading into a discrete value such as "nominal," or "off-nominal high" must be sophisticated enough to take all the environmental conditions into account. This can mean that the diagnosis problem is effectively passed off to the monitors—the diagnosis system is very simple, but relies on discrete sensor values from extremely complex monitors that diagnose the interaction between the system and its environment as part of translating continuous sensor values into discrete variables. To overcome this problem, we need to reason directly with the continuous values we receive from sensors. That is, our model needs to be a hybrid system, consisting of a set of discrete modes that the system can be in, along with a set of continuous state variables. The dynamics of the system is described in terms of a set of equations governing the evolution of the state variables, and these equations will be different in different modes. In addition, a transition function describes how the system moves from one mode to another, and an observation function defines the likelihood of an observation given the mode and the values of the system variables.

This hybrid model can be seen as a partially observable Markov decision process (POMDP) [1]. POMDPs are frequently used as a representation for decision-theoretic planning problems, where the task is to determine the best action to perform given the current estimate of the actual state of the system. This estimate, referred to as

the belief state,is exactly what we would like to determine in the diagnosis problem, and the problem of keeping the belief state updated is well understood in the decision theory literature. The belief state is a probability distribution over the system states—that is, for every state it gives the probability of being in that state, given our prior beliefs about the state of the system, and the sequence of observations and actions that have occurred so far.

Unfortunately, maintaining an exact belief state is computationally intractable for the types of problem we are interested in. Since our model contains both discrete and continuous variables, the belief state is a set of multidimensional probability distributions over the continuous state variables, with one such distribution for each mode of the system. These distributions may not even be unimodal, so just representing the belief state is a complex problem, but updating it when new observations are made is intractable for hybrid models in all but the simplest of models (see [8] for an illustration of this). Therefore, an approximation needs to be made. As we said above, we will use a *particle filter* to approximate the belief state and keep it updated.

A particle filter represents a probability distribution using a set of discrete samples, referred to as particles, each of which has an associated weight. The set of weighted particles constitutes an approximation to the belief state, and has the advantage over other approaches such as Kalman Filters [6] that it can represent arbitrary distributions. To update the distribution when a new observation is made, we treat each particle as a hypothesis about the state of the system, apply the model to it to move it to a new state, and multiply the weight of the particle by the likelihood of making the observation in that new state. To prevent a small number of particles from dominating the probability distribution, the particles are then resampled, with a new set of particles, each of weight one, being constructed by selecting samples randomly based on their weight from the old set.

Particle filters have already proven very successful for a number of tasks, including visual tracking [7] and robot navigation [4]. Unfortunately, they are less well suited to diagnosis tasks. This is because the mode transitions that we are most interested in detecting namely transitions to fault states typically have very low probability of actually occurring. Thus, there is a risk that there will be no particle in a fault state when a fault occurs, and the system will be unable to diagnose the fault. We propose a solution to this problem by thinking of a particle filter as a convenient way to divide the computation time that is available for doing diagnosis between the candidate states that the system could be in. A conventional particle filter splits the particles (and hence the computation time) according to how well the states predict the observations, but with this approach we will also spend some computation time on fault states that are important to diagnose. We do this simply by ensuring that there are always some particles in those states. As we will show, the details of the particle filter algorithm mean that we can add these additional particles without biasing the diagnosis that results.

In the next section we discuss the hybrid model of the rover in detail. In Section 3 we describe particle filtering and demonstrate its weaknesses when applied to diagnosis problems, and in Section 4 we will describe our modifications to the standard particle filter in detail. In Section 5 we present some preliminary results on real rover data, using a simple version of our proposed approach. The final section looks at the relationship between this work and some previous approaches to this problem, and discusses some future directions for this work.

2 Modeling a Planetary Rover

As we said above, we model a rover as a hybrid system. The discrete component of the rover's state represents the various operational and fault modes of the rover, while the continuous state describes the speed of the wheels, the current being drawn by various subsystems, and so on. Following [13], our rover model consists of a tuple $\langle M, V, T, E, O \rangle$ where the elements of the tuple are as follows:

- M is the set of discrete modes the system can be in. We assume that M is finite, and write m for an individual system mode.
- V is the set of variables describing the continuous state of the system.
- T is a transition function that defines how the system moves from one mode to another over time. We write $\Pr_T(m, m')$ for the probability that the system moves from mode m to mode m'. We may also include a second transition function $\Pr_T(m, a, m')$ which is used when an action a occurs. This gives the probability of moving from m to m' when action a is executed.
- E is a set of equations that describe the evolution of the continuous variables over time. The equations that apply at a given time potentially depend on the system mode, so we write E_m for the equations that apply in mode m. These equations will in general include a noise term to account for random variations in the state variables. Here we will assume Gaussian noise, with the parameters of the Gaussian determined individually for each equation.
- O is a function mapping the system state into observations. We will assume that the observable system characteristics are some subset of the system variables V, with their values corrupted by Gaussian noise (again with parameters that may be a function of the variable, and the system mode), so we write $O(v, m)$ for the observed value of some variable v in mode m.

We will also write $\Pr(s'|s)$ for the probability distribution over future states s' given some state s,where s and s' are hybrid states, so $\Pr(s'|s)$ includes both the distribution over the future mode given by $\Pr T(m'|m)$,and the distributions over the continuous variables given by E_m.

The diagnosis problem now becomes the task of determining the current mode m that the system is in, and the values of all the state variables in V (the results we will present will only show the probability distribution over discrete modes, but the algorithm produces a distribution over the full hybrid state).

The experiments we will present in Sections 3 and 5 use actual telemetry data from NASA Ames Marsokhod rover. The Marsokhod is a planetary rover built on a Russian chassis that has been used in field tests from 1993–99 in Russia, Hawaii, and the deserts of Arizona and California. The rover has six independently driven wheels, and for the experiments we present here, the right rear wheel had a broken gear, and so rolls passively. The Marsokhod has a number of sensors, but we will restrict our attention to diagnosing the state of the broken wheel, and will therefore use only data from the wheel current and wheel odometry sensors. We will treat each wheel independently in the diagnosis. For each wheel, we have a model, taken from [13], with the following characteristics:

- M consists of 23 system modes of which 14 are fault states.
- V consists of variables for the wheel current and wheel speed, and the derivatives of current and speed.
- T is a fairly sparse matrix, with at most six successors for any given mode. The probability of a transition to a fault state is 0.01 or less. All commands are described by one transition function

for the start and one for the end of a command because the data doesn't identify which command occurred.
- The state equations in E consist of the previous value plus a constant term and noise. The noise is Gaussian with standard deviation in the range 0.001 to 1.0, and the equations are independent for each state variable.
- The equations in O are independent for each variable (but vary depending on the mode), and include a Gaussian noise term with a standard deviation that varies from 0.001 to 1.0.

3 Particle Filters

A particle filter approximates an unknown probability distribution using a weighted set of samples. Each sample or particle consists of a value for every state variable, so it describes one possible complete state the system might be in. As observations are made, the transition function is applied to each particle individually, moving it stochastically to a new state, and then the observations are used to re-weight each particle to reflect the likelihood of the observation given the particle's new state. In this way, particles that predict the observed system performance are highly weighted, indicating that they are in likely states of the system. A major advantage of particle filters is that their computational requirements depend only on the number of particles, not on the complexity of the model. This is of huge importance to us as it allows us to do diagnosis in an anytime fashion; increasing or decreasing the number of particles depending on the available computation time.

To implement a particle filter, we require three things:

- A probability distribution over the initial state of the system.
- A model of the system that can be used to predict, given the current state according to an individual particle, a possible future state of that particle. Since T is stochastic, and E includes noise terms, the predictive model selects a new state for the particle in a Monte Carlo fashion [5], choosing by sampling from the probability distribution over possible future states.
- A way to compute the likelihood of observing particular sensor values given a state. In our case, this is given by the observation function O.

The particle filtering algorithm is given in Figure 1. Step (i) is the *predictive step*, where a new state is calculated in a Monte Carlo way for each particle, and this new state is then conditioned on the observations in step (ii) (we call this the *re-weighting step*). The predictive step is performed by applying T to each particle, and then applying the appropriate equations from E to the state variables, sampling values from the Gaussian error terms. Once the particles have been re-weighted, we can then calculate the probability of each mode simply by summing the weights of the particles in the mode. We refer to step (b) as the *resampling step*. For more details on the properties of particle filters see e.g. [3].

3.1 Problems with Particle Filters for Diagnosis

Unfortunately, there are a number of difficulties in applying particle filters to diagnosis problems. In particular, the filter must have a particle in a particular state before the probability of that state can be evaluated. If a state has no particles in it, its probability of being the true state of the system is zero. This is a particular problem in diagnosis problems because the transition probabilities to fault states are typically very low, so particles are unlikely to end up in fault states

1. Create a set of n particles where each particle p_i has a state s_i and a weight w_i. s_i is sampled randomly from the prior state distribution, and $w_i = 1$.
2. For each time step, do:
   (a) Replace each particle p_i with p'_i as follows:
      i. Select a future state s'_i by sampling from $\Pr(s'_i|s_i)$, the distribution over possible future states given by the model.
      ii. Re-weight p_i by multiplying its weight by the probability of the observations o given s'_i as follows:

$$w'_i = \Pr(o|s'_i)w_i$$

   (b) Resample n new particles $p_1, \ldots, p_n$ by copying the p' current particles where each particle p'_j is added to the new samples with the following probability:

$$\Pr(p_i = p'_j) = \frac{w'_j}{\sum_{k=1}^{n} w'_k}$$

Figure 1. The particle filtering algorithm.

during the Monte Carlo predictive step. This situation is known as *sample impoverishment*.

Figure 2 illustrates this problem. Each graph shows the most likely modes that the wheel is in (the y-axis is the total weight of the particles in each mode, so a value of 10000 implies that all particles are in that mode), shown over part of one of the trials in which the wheel is initially idle, and then at step 12 is commanded to drive forward at a fixed speed. The graphs on the left show the performance of Wheel 1, which is operating nominally. The graphs on the right show the performance of Wheel 6, which is faulty. In the top line, the probability of the fault occurring is 0.1 rather than its true value of 0.01. Here the fault is quickly detected in Wheel 6. In the bottom line of graphs, the fault probability is set to its true value, and in this case the fault is not successfully detected because insufficient particles enter the fault state. One might expect that once a particle enters the fault state its weight would be high since it would predict well, and at the resampling step it should lead to several new particles being created. Unfortunately, this did not occur in this situation because although some particles did enter the fault state, their continuous parameter values did not agree with the observations well, so they still had low weights. The continuous parameters did not match because each of the particles that entered the fault state came from the COMMANDEDRUNNING state, in which the current and wheel speed are expected to be much higher than the observed values.

4 Importance Sampling

The simplest solution to the sample impoverishment problem is to increase the number of particles being used. Given the constraints imposed on on-board systems, this approach is probably unrealistic. The data presented above was implemented in Java, using 10,000 particles per wheel, and runs in approximately 0.5 seconds per update on a 750MHz Pentium 3. This is probably at the upper limit of the number of particles we could expect to use on-board a rover as the time available for diagnosis is longer, but there will be less computation devoted to diagnosis. Thus running with ten times as many particles (which is roughly equivalent to multiplying the fault

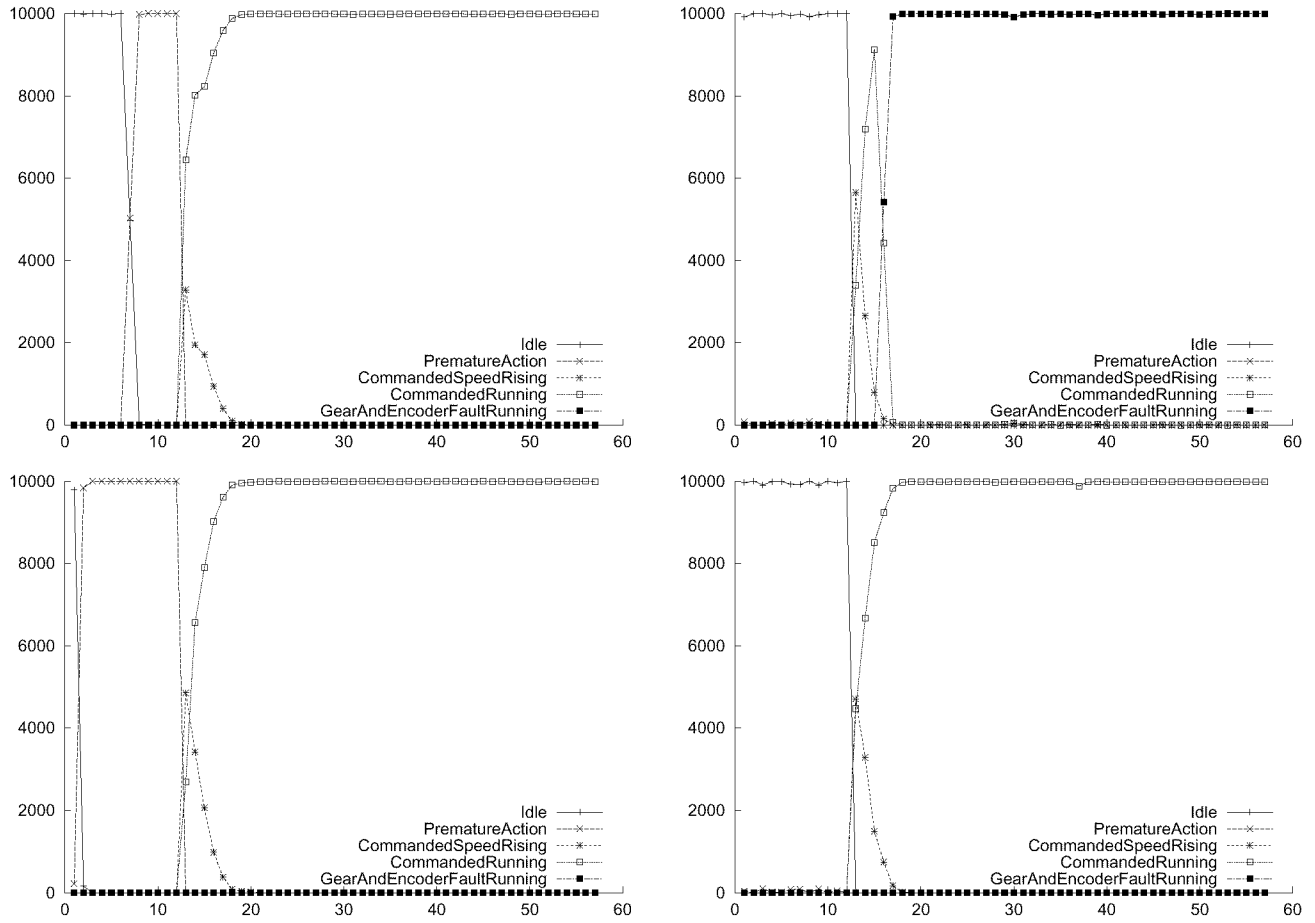

Figure 2. Results for Wheel 1 (left side) and Wheel 6 (right side). In the top row, the probability of a fault is ten times its true value, while the bottom row uses the true probabilities. The fault (GEARANDENCODERFAULTRUNNING) in Wheel 6 is quickly detected in the top row, but is never discovered in the bottom row due to sample impoverishment.

probability by ten) is probably impractical on the rover, and even 10,000 particles may be unrealistic as the model gets more complex. This could be somewhat overcome by only increasing the number of particles when there is some evidence that the system is predicting poorly. In order to achieve this, we need some measure of when this occurs. The obvious measure is to look at the total weight of the particles after conditioning on the observations. If no particles are predicting the observations well the total weight should drop. Unfortunately, in practice this is rarely useful because there are a number of other possible causes for this behavior. For example, particles moving from a state in which there is high confidence in the sensor readings to a state with more sensor noise will tend to drop in weight even if they are still predicting the observations well. We see this in the Marsokhod model because the IDLE mode has relatively large variance for the observation noise, whereas the COMMANDEDRUNNING mode has smaller variance, so the total particle weight increases when the system moves from the IDLE to the COMMANDEDRUNNING mode, even for wheel 6 where COMMANDEDRUNNING predicts the observations poorly.

Another way to reduce the likelihood of sample impoverishment is to take advantage of some results from *importance sampling* (see e.g. [9]). In importance sampling, we want to sample from some distribution P, but we can't. Instead, we sample from some other distribution Q, and weight each sample s by $\Pr_P(s)/\Pr_Q(s)$, the ratio of the likelihood of sampling s from P to the likelihood of sampling it from Q. The weighted sample is then an unbiased sample from P, as long as Q is non-zero everywhere that P is non-zero. In fact, importance sampling is exactly what the particle filter algorithm is doing. For a particle filter, the unknown distribution P is the posterior distribution we are trying to compute, Q is the prior distribution (the set of samples before the observation), and the re-weighting step corresponds to the importance sampling weight computation.

Given that whatever we choose for Q, the weighted samples are an unbiased sample from P, we can add arbitrary samples to our particle filter at the resampling step, and still end up with an unbiased posterior distribution. We will use this property to ensure that we have samples in the system modes that are important to us (hence the name importance sampling). The question then is how to choose Q. We can imagine an oracle that provides a set of candidate states the system might end up in, given the current distribution over state. When we resample, we simply make sure that there are always some particles in the states provided by the oracle. If those states explain the subsequent observations well, the particles in them will get high weight, and are likely to be resampled with more particles at the next

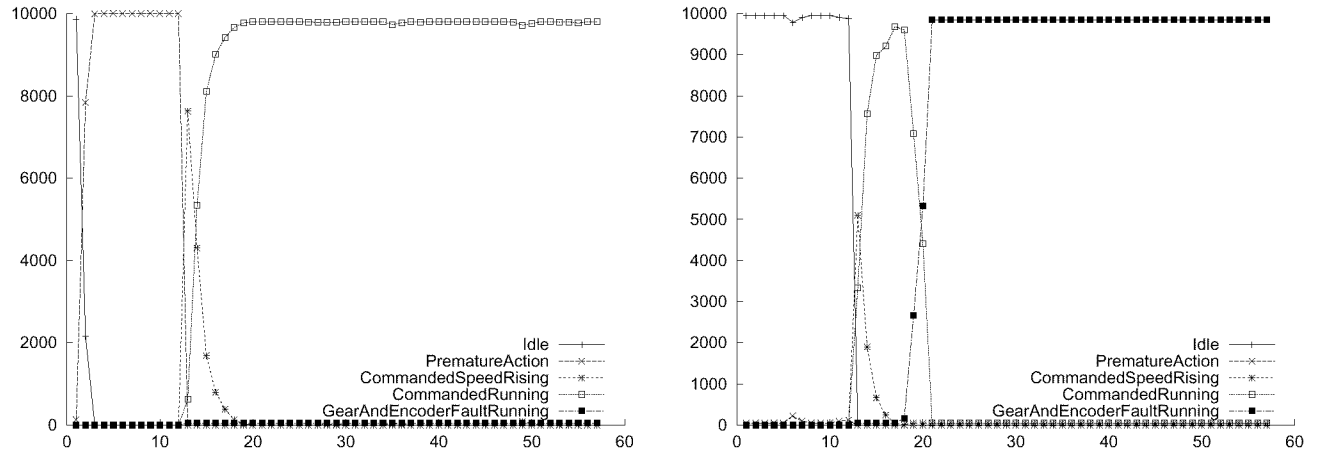

Figure 3. Results for the importance sampled particle filter. All states with $> 25\%$ probability were used as starting points for the forward search, and 0.5% of the particles were assigned to each of the found states. On the left are results for wheel 1, and on the right for wheel 6.

step. On the other hand, if they predict the observations poorly, the particles will quickly disappear again.

The question that remains is how to implement the oracle. For a complex system such as a planetary rover, with many components each with its own set of possible failure modes, there are exponentially many possible failure modes, so this is a non-trivial problem. However, one approach that seems promising is to use a traditional model-based diagnosis system such as Livingstone [14]. These discrete systems typically operate more quickly than hybrid approaches, so can be used to suggest hypotheses without significantly affecting computation time. We pointed out in the introduction that they are not in general suitable for diagnosing rovers, but they could be used to identify sets of likely system modes for the hybrid system to be used in. The integration of Livingstone with the particle filter approach is currently work in progress, as it adds a number of additional complications including building an additional system model, and ensuring that the discrete and hybrid models agree with one another and can easily be translated back and forth.

For simpler systems such as the Marsokhod wheel diagnosis we have used in this paper, the above approach is unnecessary. Instead, we can use an oracle based on forward search from the current high-probability states. Since each system mode in this model has at most six possible successors, and there are typically only two to three high probability modes at any time, we find in practice that in most cases a simple one-step look-ahead search adds fewer than five modes to those that already contain particles.

5 Results

The results we present here are based on the Marsokhod model we described in Section 2. Dr. Rich Washington supplied the model and the data, which came from his work on using Kalman filters for rover diagnosis [13]. The only changes made to the model were to make it suitable for use with a particle filter; no changes were made to model parameters or transition probabilities. To demonstrate our approach we use a small piece of one of the telemetry data files (the same piece used in Section 3) in which the rover is initially idle, and then a drive command is issued, resulting in an increase in current to each wheel, followed by a corresponding increase in speed, and then a constant speed. As before, wheel 6 is faulty, with a broken gear (this corresponds to the GEARANDENCODERFAULTRUNNING state in the model).

Figure 3 shows the results for the importance sampled particle filter. We used single step forward search from all states with probability > 0.25 to select the set of bias states. Each of these states was then guaranteed to receive at least 0.5% of the total number of particles at each re-sampling step. The left hand graph is the probable states for Wheel 1, as before. Like the graph in the bottom row of Figure 2, the PREMATUREACTION state was given high probability before step 13. This state appears where the effects of an action are seen before the signal to perform the action, due to problems with the rover telemetry. In this case it is a spurious result due to the model of the IDLE state not allowing sufficient noise in the observations. A small adjustment to the model would remove this problem, which is only present in the data for two of the wheels. The right hand graph shows the same data for Wheel 6. In this case, the fault state dominates the probability distribution after step 20, seven steps after the command to drive the wheel was observed, as compared with three steps for the model with increased fault probabilities (Figure 2).

6 Discussion and Relation to Other Work

An important thing to note is that standard particle filters treat the model essentially as a black box, using it only to predict future states of the system. We have described one approach that exploits the structure in the model to make diagnosis using particle filters more effective. This is by no means the only way to exploit that structure, and we are in the process of looking at other techniques. Possibilities include making a single-fault assumption (but relaxing it if it predicts the observations poorly), and taking advantage of independence between components in the system to reduce the number of samples needed, or even to diagnose subsystems independently.

One closely related piece of work is Verma et al.'s decision-theoretic particle filter [11]. Their approach is similar to ours, but they assign a utility—which corresponds to how important each state is to diagnose—to every state and multiply the probability of a transition by the utility of the state that results. This alters the transition function to favor important states, rather like the approach we took

in Figure 2. For relatively simple diagnosis tasks such as the one we have presented here, the approaches seem very similar. However, designing a utility function to produce the right diagnoses, without causing too many false diagnoses of faults is a difficult task, especially as any reasonable utility function would give all fault states a high utility. In [10], they refine this approach, again using a *risk function* that scores states by how important it is to diagnose them correctly, but this time modifying the particle filter algorithm so that the samples are distributed according to the product of the posterior probability distribution and the risk factor. This ensures that samples in high-risk states have higher weights, and the true posterior can be recovered from the risk-sensitive posterior, but still suffers from the problem of sample impoverishment. These approaches are orthogonal to the approach described here, and we are currently working on combining the two.

Another related effort is the work of Washington [13] that applies Kalman Filters to this problem. In this work, the continuous dynamics in each mode is tracked by a set of Kalman filters. The main problem with the approach is that the number of filters tends to increase over time because each time a transition is made to a state the initial conditions for the filter are different, and filters with different initial conditions cannot be combined. This is not a problem for particle filter-based approaches because the particle filters can represent arbitrary distributions over the parameter values, so particles entering a state with two different sets of initial conditions will form a bi-modal distribution. As we said above, we used the model and data from this paper in our own work. We see fewer errors in the mode identification with our approach than in Washington's paper, although we are sometimes slower to identify the fault, and our computational requirements are somewhat higher.

As we said in the introduction, this paper is intended as a proof of concept. There is still much work to do on the problem of how to integrate a model from Livingstone with this system to act as an oracle. We have demonstrated that a simple look-ahead search performs quite well, but this is clearly inadequate for large diagnosis problems, particularly as most faults can occur at any time, so the one step lookahead is unlikely to scale to very large problem. We are also examining a number of other approaches to improving diagnosis with particle filters, such as backtracking when prediction is poor, and re-sampling past states based on observations that occurred more recently. Finally, we are investigating how a diagnosis system of this type would fit with the CLARAty rover architecture [12] being used for future NASA missions to Mars.

REFERENCES

[1] Dimitri P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*, Prentice-Hall, Englewood Cliffs, 1987.

[2] A. Doucet, 'On sequential simulation-based methods for Bayesian filtering', Technical Report CUED/F-INFENG/TR.310, Department of Engineering, Cambridge University, (1998).

[3] *Sequential Monte Carlo in Practice*, eds., Arnaud Doucet, Nando De Freitas, and Neil Gordon, Springer-Verlag, 2001.

[4] Dieter Fox, Wolfram Burgard, and Sebastian Thrun, 'Markov localization for mobile robots in dynamic environments', *Journal of Artificial Intelligence Research*, **11**, 391–427, (1999).

[5] *Markov Chain Monte Carlo in Practice*, eds., W. R. Gilks, S. Richardson, and D. J. Spigelhalter, CRC Press, 1996.

[6] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice*, Prentice Hall, 1993.

[7] M. Isard and A. Blake, 'CONDENSATION: Conditional density propagation for visual tracking', *International Journal of Computer Vision*, (1998).

[8] Uri Lerner, Ron Parr, Daphne Koller, and Gautam Biswas, 'Bayesian fault detection and diagnosis in dynamic systems', in *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, (2000).

[9] B. D. Ripley, *Stochastic Simulation*, Wiley, New York, 1987.

[10] Sebastian Thrun, John Langford, and Vandi Verma, 'Risk sensitive particle filters', in *Neural Information Processing Systems (NIPS)*, (December 2001).

[11] V. Verma, J. Langford, and R. Simmons, 'Non-parametric fault identification for space rovers', in *International Symposium on Artificial Intelligence and Robotics in Space (iSAIRAS)*, (2001).

[12] R. Volpe, I. A. D. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das, 'The CLARAty architecture for robotic autonomy', in *Proceedings of the 2001 IEEE Aerospace Conference*, Big Sky, Montana, (March 2001).

[13] Rich Washington, 'On-board real-time state and fault identification for rovers', in *Proceedings of the IEEE International Conference on Robotics and Automation*, (April 2000).

[14] Brian C. Williams and P. Pandurang Nayak, 'A model-based approach to reactive self-configuring systems', in *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference*, pp. 971–978, Portland, Oregon, (1996). AAAI Press / The MIT Press.